BAB 2
LANDASAN TEORI

2.1 Serangan Domain Generation Algorithm (DGA)

Serangan berbasis *Domain Generation Algorithm* (DGA) merupakan teknik yang umum digunakan oleh *malware* untuk mempertahankan persistensi komunikasi dengan server pengendali (*Command and Control* atau C&C). Pada mekanisme ini, *malware* tidak bergantung pada satu nama domain statis, melainkan menghasilkan ribuan nama domain semu secara otomatis dan periodik menggunakan algoritma tertentu. Pola dinamis ini menyebabkan jalur komunikasi *malware* sulit diputus hanya dengan metode pemblokiran daftar hitam (*blacklist*) konvensional [8, 9].

Secara teknis, penyerang menanamkan modul DGA (berbasis waktu atau benih acak) ke dalam biner *malware*. Ketika perangkat korban terinfeksi, *malware* akan membangkitkan daftar calon domain dan mencoba melakukan resolusi DNS satu per satu hingga menemukan domain yang telah didaftarkan aktif oleh penyerang [8]. Metode ini menciptakan asimetri beban kerja; penyerang cukup mendaftarkan sedikit domain, sementara sistem keamanan harus memindai ribuan domain yang dihasilkan [9]. Jika satu domain diblokir, *malware* akan segera beralih ke domain lain dalam daftar antrean, menjamin kelangsungan operasi botnet [9].

Dampak dari serangan DGA sangat signifikan terhadap operasional keamanan siber. Pertama, DGA meningkatkan ketahanan (*resilience*) infrastruktur botnet terhadap upaya *takedown* [9]. Kedua, lonjakan jumlah domain acak membebani sistem pemantauan lalu lintas jaringan, membuat proses identifikasi ancaman menjadi lebih kompleks dan memakan sumber daya [10]. Kegagalan dalam mendeteksi kanal komunikasi ini dapat berujung pada eksfiltrasi data sensitif, penyebaran muatan berbahaya lanjutan (*payload delivery*), atau serangan DDoS terkoordinasi [8].

Oleh karena itu, paradigma pertahanan kini beralih dari pendekatan reaktif berbasis *signature* menuju pendekatan prediktif berbasis *Machine Learning* (ML) dan *Deep Learning* (DL). Pendekatan ini bertujuan mengenali karakteristik intrinsik domain DGA seperti pola keacakan karakter atau struktur linguistik yang tidak wajar sehingga dapat mendeteksi ancaman baru yang belum pernah terlihat sebelumnya (*zero-day*) [9, 10].

2.1.1 Kebutuhan Deteksi DGA dibanding Indikator DNS Berbasis Trafik

Secara umum, *Domain Generation Algorithm* (DGA) membuat *malware* menghasilkan banyak kandidat nama domain secara periodik (misalnya harian) berdasarkan *seed* tertentu, seperti waktu sistem. Korban kemudian melakukan resolusi DNS terhadap kandidat tersebut hingga menemukan domain yang didaftarkan penyerang pada periode yang sama [1]. Pola ini menghasilkan banyak kueri gagal (misalnya *NXDOMAIN*) dan membuat strategi pemblokiran berbasis *blacklist* menjadi kurang efektif karena domain berubah-ubah dan jumlahnya sangat besar [1, 9].

Dalam pertahanan, indikasi DGA umumnya dianalisis melalui dua pendekatan yaitu telemetri DNS berbasis trafik (misalnya rasio *NXDOMAIN* atau pola kueri tidak lazim atau*unusual DNS queries*) dan juga analisis string nama domain (*domain-only*) [9]. Pendekatan telemetri membutuhkan log DNS yang lengkap dan stabil, serta dapat dipengaruhi *resolver*, *caching*, dan pola penggunaan jaringan. Sebaliknya, pendekatan *domain-only* memanfaatkan ciri leksikal domain seperti panjang, entropi, dan pola *n-gram* sehingga lebih mudah diterapkan ketika data trafik tidak tersedia. Karena itu, penelitian ini memfokuskan deteksi pada *Second Level Domain* (SLD).

Perbandingan ringkas antara pendekatan telemetri DNS dan pendekatan berbasis nama domain ditunjukkan pada Tabel 2.1.

Tabel 2.1. Perbandingan deteksi DGA berbasis telemetri DNS dan berbasis nama domain

Aspek	**Telemetri DNS (*traffic-based*)**	**Nama domain (*domain-only*)**
Kebutuhan data	Log DNS (misalnya waktu, *resolver*, respons)	String SLD saja
Kelebihan	Menangkap konteks perilaku kueri	Ringan, portabel, bisa untuk *pre-filtering*
Keterbatasan	Dipengaruhi *caching* dan pola jaringan	Tidak memuat konteks perilaku DNS
Contoh indikator	Rasio *NXDOMAIN*, periodisitas kueri	Entropi, panjang, distribusi karakter, *n-gram*

2.2 Klasifikasi dan Deteksi DGA berbasis Pembelajaran Mesin

Untuk mengatasi keterbatasan metode statis, penelitian modern memanfaatkan algoritma pembelajaran mesin untuk mengklasifikasikan nama domain. Pendekatan ini terbagi menjadi dua kategori utama: berbasis fitur (*Feature-based*) dan berbasis sekuensial (*Sequence-based*).

Pendekatan berbasis fitur, seperti Random Forest, bekerja dengan mengekstraksi atribut statistik dari nama domain (misalnya panjang string, rasio konsonan, dan entropi) untuk melatih *classifier* [8]. Random Forest sering

menjadi pilihan utama karena kemampuannya menangani data berdimensi tinggi dan ketahanannya terhadap *overfitting* melalui mekanisme *ensemble* [8].

Di sisi lain, pendekatan *Deep Learning* memungkinkan pembelajaran representasi otomatis langsung dari data mentah. Model sekuensial seperti *Long Short-Term Memory* (LSTM) dan *Bidirectional LSTM* (BiLSTM) mampu mempelajari dependensi antar karakter dalam nama domain tanpa memerlukan rekayasa fitur manual yang ekstensif [11]. BiLSTM secara khusus unggul karena kemampuannya memproses urutan karakter dari dua arah, memberikan pemahaman konteks yang lebih komprehensif dibandingkan LSTM standar [11]. Kombinasi kedua pendekatan ini (Hybrid) juga mulai dieksplorasi untuk menggabungkan keunggulan interpretasi fitur statistik dengan kekuatan ekstraksi pola saraf tiruan [8].

2.3 Pra-pemrosesan Data

Pra-pemrosesan data dilakukan untuk memastikan dataset yang digunakan dalam penelitian ini memiliki format yang konsisten, bebas entri tidak valid, dan tidak mengandung duplikasi yang berpotensi mengganggu proses pelatihan maupun evaluasi model. Dalam pemrosesan teks, tahapan pembersihan seperti penyeragaman huruf dan perapihan spasi digunakan agar representasi data lebih stabil dan mengurangi variasi yang tidak bermakna pada string [12]. Pada konteks deteksi domain DGA, pra-pemrosesan juga umum diarahkan untuk memfokuskan analisis pada bagian nama domain yang paling informatif, yaitu komponen inti yang merepresentasikan pola pembentukan domain [13, 14]. Sejalan dengan batasan masalah penelitian ini, pra-pemrosesan difokuskan pada karakter ASCII atau Latin dan dianalisis pada level *second-level domain* (SLD), tanpa melibatkan inspeksi jaringan dinamis.

2.3.1 Penggabungan Data dan Pelabelan

Dataset pada penelitian ini dibangun dari unifikasi beberapa sumber, sehingga diperlukan penggabungan data agar kelas domain *legit* dan DGA berada dalam satu kerangka yang seragam. Pendekatan penggabungan dari beberapa sumber juga digunakan dalam studi deteksi domain terbangkitkan algoritmik, di mana domain *benign* dan domain berbahaya dikumpulkan dari sumber yang berbeda untuk membentuk dataset pelatihan yang lebih representatif [13]. Pada

implementasi penelitian ini, data disusun menjadi dua berkas utama, yaitu berkas domain *legit* dan berkas domain DGA, kemudian masing-masing diberi label biner: label 0 untuk *legit* dan label 1 untuk DGA, sebelum akhirnya digabung menjadi satu dataset terpadu untuk tahap berikutnya.

2.3.2 Normalisasi dan Pembersihan String Domain

Normalisasi dilakukan untuk menyeragamkan bentuk teks domain sebelum ekstraksi fitur. Berdasarkan praktik umum pra-pemrosesan teks, dua langkah awal yang penting adalah menghapus spasi tidak perlu pada awal atau akhir string dan menyeragamkan huruf menjadi huruf kecil agar perbedaan kapitalisasi tidak menghasilkan entri yang dianggap berbeda [12]. Implementasi penelitian ini menerapkan *strip* untuk membersihkan spasi tepi dan *lowercasing* untuk memastikan domain berada dalam format yang konsisten, sehingga proses seperti deduplikasi dan ekstraksi SLD dapat dilakukan dengan lebih andal.

2.3.3 Ekstraksi *Second-Level Domain* (SLD)

Ekstraksi SLD bertujuan memfokuskan analisis pada inti nama domain dan menghilangkan komponen yang relatif kurang informatif seperti ekstensi TLD. Studi Wang dkk. menegaskan bahwa TLD bersifat "hampir tidak informatif" untuk klasifikasi, sehingga komponen TLD dihapus pada tahap pra-pemrosesan [13]. Karena penelitian ini juga membatasi analisis hanya pada SLD, maka komponen TLD dan subdomain dieliminasi agar pola pembentukan string utama menjadi pusat perhatian model. Pada implementasi, ekstraksi SLD dilakukan melalui pemisahan komponen domain, dengan opsi metode yang lebih akurat menggunakan daftar sufiks publik melalui pustaka *tldextract* ketika diperlukan, agar penanganan TLD bertingkat lebih tepat.

2.3.4 Pemeriksaan Nilai Kosong dan Entri Tidak Valid

Setelah normalisasi dan ekstraksi SLD, dilakukan pemeriksaan entri kosong dan nilai hilang untuk menghindari masalah pada tahap berikutnya. Pada data yang berasal dari banyak sumber, entri kosong dapat muncul akibat ketidakkonsistenan format atau baris yang tidak lengkap. Karena penelitian ini mengandalkan string SLD sebagai unit analisis, maka entri yang tidak memiliki SLD yang valid perlu dibuang agar tidak menimbulkan noise pada pelatihan model.

2.3.5 Deduplikasi dan Pemeriksaan Konflik Label

Penggabungan dari banyak sumber meningkatkan risiko munculnya domain yang sama lebih dari satu kali. Duplikasi semacam ini perlu dihapus karena dapat memunculkan bias evaluasi, terutama jika domain yang identik muncul pada data latih dan data uji, sehingga performa model tampak lebih tinggi dari kondisi sebenarnya. Literatur terkini menunjukkan bahwa deduplikasi sederhana merupakan langkah penting untuk meningkatkan kualitas data dan mencegah dampak negatif dari entri yang berulang pada analisis berbasis komputasi [15]. Pada implementasi penelitian ini, deduplikasi dilakukan pada level SLD agar setiap SLD hanya direpresentasikan satu kali dalam dataset akhir. Selain itu, dilakukan pemeriksaan konflik label, yaitu kondisi ketika domain yang sama muncul dengan label berbeda akibat perbedaan sumber, sehingga kasus semacam ini dapat diidentifikasi dan ditangani agar konsistensi label tetap terjaga.

2.4 Metode *Hold-Out* untuk Pembagian Data Latih dan Data Uji

Setelah dataset bersih dan terlabel tersedia, penelitian ini menerapkan metode *hold-out* untuk memisahkan data menjadi bagian pelatihan dan pengujian. Metode *hold-out* membagi dataset menjadi dua subset, yaitu subset pelatihan untuk membangun model dan subset pengujian yang disimpan terpisah untuk mengukur kemampuan generalisasi model pada data yang belum pernah dilihat [16]. Dalam penelitian ini, *hold-out* digunakan sebagai pemisahan awal agar evaluasi akhir dilakukan pada data uji yang benar-benar independen. Dengan demikian, metrik seperti akurasi, presisi, *recall*, *F1-score*, dan ROC-AUC mencerminkan performa model Random Forest dan BiLSTM pada domain baru di luar data latih.

Pada implementasi, pembagian dilakukan dengan skema 80:20 dan menggunakan *stratification* (*stratify*) sehingga proporsi kelas DGA dan legit pada data latih serta data uji tetap terjaga. Strategi ini penting untuk mencegah evaluasi yang bias akibat komposisi kelas yang berbeda secara ekstrem antara subset pelatihan dan pengujian, terutama pada dataset yang berpotensi *imbalanced*.

2.5 Ketidakseimbangan Kelas dan Penyeimbangan Data dengan *Random Undersampling*

Pada klasifikasi biner, ketidakseimbangan kelas/*class imbalance* terjadi ketika jumlah sampel pada satu kelas jauh lebih dominan dibanding kelas lainnya.

Kondisi ini sering dinyatakan melalui rasio ketimpangan/*imbalance ratio* (IR), yaitu perbandingan jumlah sampel kelas mayoritas terhadap kelas minoritas ($IR = N_{maj}/N_{min}$). Semakin besar nilai IR, semakin kuat kecenderungan model untuk memprioritaskan kelas mayoritas, sehingga performa pada kelas minoritas dapat turun walaupun akurasi keseluruhan terlihat tinggi [17, 18]. Dalam konteks deteksi domain berbahaya, ketidakseimbangan kelas juga sering muncul karena sumber data yang berbeda dapat menghasilkan jumlah sampel yang tidak proporsional antara domain *benign* dan domain *malicious*; karena itu, banyak penelitian memasukkan tahap mitigasi ketimpangan agar model lebih peka terhadap kelas yang lebih jarang [19].

Salah satu pendekatan pada level data adalah penyeimbangan melalui *random undersampling*, yaitu mengurangi jumlah sampel pada kelas mayoritas dengan cara memilih dan menghapus sebagian sampel secara acak sampai distribusi kelas menjadi lebih seimbang [17]. Keunggulan metode ini adalah sederhana dan cepat, serta tidak membuat data sintetis baru. Namun, konsekuensinya adalah potensi hilangnya informasi karena sebagian sampel kelas mayoritas dibuang; oleh sebab itu, penerapannya perlu dirancang agar tetap menjaga kemampuan generalisasi model [18]. Studi komparatif pada berbagai teknik penanganan ketimpangan juga menunjukkan bahwa *random undersampling* dapat meningkatkan sensitivitas/recall pada kelas minoritas pada beberapa skenario, meskipun efek akhirnya tetap bergantung pada karakteristik data dan model yang digunakan [20].

Pada penelitian ini, penyeimbangan dilakukan mengikuti praktik evaluasi yang menjaga keadilan pengujian, yaitu hanya diterapkan pada data latih setelah pemisahan *hold-out* 80:20. Artinya, distribusi pada data uji 20% tetap merepresentasikan kondisi asli dataset, sedangkan data latih 80% diseimbangkan menggunakan *random undersampling* dengan mereduksi kelas mayoritas pada data latih hingga setara dengan kelas minoritas. Strategi ini bertujuan mengurangi bias model ke kelas dominan saat pelatihan, sehingga Random Forest maupun BiLSTM yang dilatih dapat lebih stabil dalam membedakan pola domain DGA dan domain legit pada pengujian akhir.

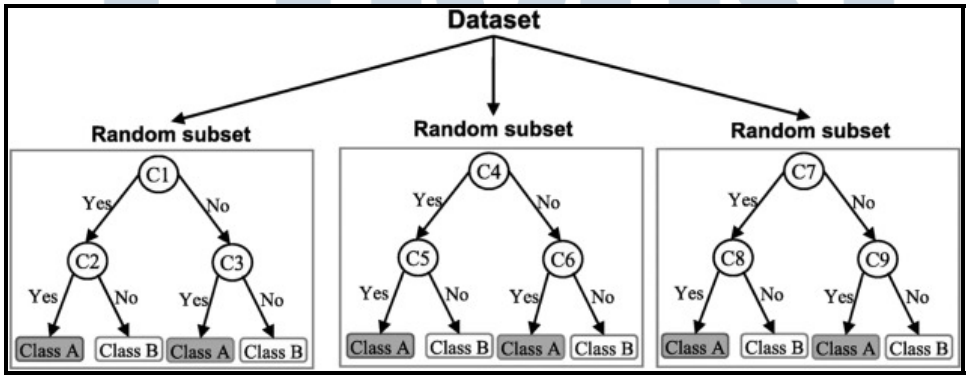
2.6 Random Forest

2.6.1 Konsep Dasar

Random Forest adalah algoritma *ensemble learning* yang terdiri dari sekumpulan pohon keputusan (*Decision Trees*). Prinsip utamanya adalah *Bagging* (*Bootstrap Aggregating*), di mana setiap pohon dilatih menggunakan subset data yang diambil secara acak dengan pengembalian, dan setiap pemecahan (*split*) node hanya mempertimbangkan subset fitur acak [21]. Strategi ini bertujuan menciptakan keragaman (*diversity*) antar pohon untuk meredusi varians dan risiko *overfitting* yang sering terjadi pada pohon keputusan tunggal [22].

Keputusan klasifikasi akhir diambil berdasarkan suara terbanyak (*Majority Voting*) dari seluruh pohon dalam hutan. Contoh sederhana prediksi Misalkan sebuah domain menghasilkan fitur leksikal seperti panjang SLD = 18, entropi = 3.9, rasio vokal = 0.11, dan jumlah digit = 5. Sebuah *Random Forest* berisi banyak pohon keputusan akan menguji kombinasi fitur tersebut pada masing-masing pohon. Misalnya, sebagian pohon memutuskan kelas DGA karena entropi tinggi dan rasio vokal rendah, sementara sebagian lain memutuskan *legit* karena fitur tertentu tidak ekstrem. Prediksi akhir ditentukan melalui *majority voting*, yaitu kelas yang paling banyak dipilih oleh pohon-pohon tersebut.

Gambar 2.1 memperlihatkan ilustrasi struktur *Random Forest*, yaitu sekumpulan pohon keputusan yang digabungkan melalui mekanisme *majority voting*.



Gambar 2.1. Arsitektur algoritma *Random Forest*.

Secara matematis, prediksi kelas $\hat{y}$ didefinisikan sebagai:

$$\hat{y} = \text{mode}\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_B(\mathbf{x})\}, \tag{2.1}$$

dengan keterangan $h_b(\mathbf{x})$ adalah prediksi dari pohon ke-b dan B adalah jumlah total pohon.

2.6.2 Kriteria Pemisahan (Gini Impurity)

Dalam konstruksi setiap pohon, pemilihan fitur terbaik untuk memecah node didasarkan pada penurunan ketidakmurnian (*impurity*).

Metrik yang digunakan adalah *Gini Impurity*, yang dihitung dengan persamaan:

$$Gini(t) = 1 - \sum_{k=1}^{K} p(k|t)^2, \tag{2.2}$$

dengan $p(k|t)$ adalah proporsi sampel pada kelas ke-k di node t dan K adalah jumlah kelas. Karena penelitian ini merupakan klasifikasi biner, maka $K = 2$ untuk kelas legit dan DGA.

2.6.3 Aplikasi pada Deteksi DGA

Dalam konteks deteksi DGA, Random Forest terbukti efektif karena kemampuannya menangkap pola non-linear dari fitur leksikal domain. Studi komparasi menunjukkan bahwa Random Forest kerap mengungguli algoritma lain seperti Naive Bayes atau SVM dalam hal akurasi dan stabilitas, terutama ketika dihadapkan pada variasi fitur yang beragam dari berbagai keluarga DGA [23].

2.7 Bidirectional LSTM (BiLSTM)

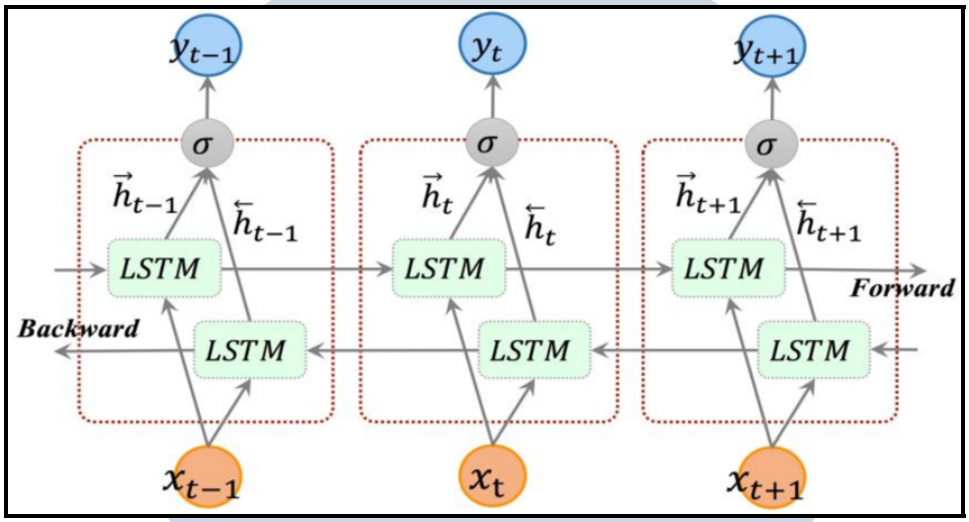
2.7.1 Arsitektur LSTM dan Gerbang Logika

Long Short-Term Memory (LSTM) adalah varian dari *Recurrent Neural Network* (RNN) yang dirancang untuk mengatasi masalah *vanishing gradient* pada data sekuensial panjang. Inti dari LSTM adalah sel memori yang diatur oleh mekanisme gerbang (*gates*) untuk mengontrol aliran informasi [11]:

Gerbang pada LSTM terdiri dari:

- Gerbang lupa (*forget gate*, f_t): menentukan informasi lama yang dibuang.
- Gerbang masukan (*input gate*, i_t): menentukan informasi baru yang disimpan.
- Gerbang keluaran (*output gate*, o_t): menentukan keluaran berdasarkan kondisi memori saat ini.

Gambar 2.2 menunjukkan arsitektur *Bidirectional LSTM* (BiLSTM) yang memproses sekuens dari dua arah untuk menangkap dependensi karakter kiri-ke-kanan dan kanan-ke-kiri.



Gambar 2.2. Arsitektur *Bidirectional LSTM* (BiLSTM). Diadaptasi dari [2].

Persamaan pembaruan sel LSTM pada waktu t adalah sebagai berikut:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{2.3}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2.4}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{2.5}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \tag{2.6}$$

dengan keterangan x_t adalah input saat ini, h_{t-1} adalah *hidden state* sebelumnya, dan C_t adalah *cell state*.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.7}$$

$$h_t = o_t * \tanh(C_t), \tag{2.8}$$

Pada penelitian ini, x_t merepresentasikan token karakter dari SLD yang telah dipadatkan (*padding*) menjadi panjang tetap sebelum diproses oleh BiLSTM.

2.7.2 Mekanisme Bidirectional

BiLSTM memperluas arsitektur ini dengan memproses urutan data dalam dua arah: maju (*forward*) dari awal ke akhir, dan mundur (*backward*) dari akhir ke

awal [2].

$$\overrightarrow{\mathbf{h}}_t = \text{LSTM}_{fwd}(x_t, \overrightarrow{\mathbf{h}}_{t-1}), \tag{2.9}$$

$$\overleftarrow{\mathbf{h}}_t = \text{LSTM}_{bwd}(x_t, \overleftarrow{\mathbf{h}}_{t+1}), \tag{2.10}$$

$$\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t \oplus \overleftarrow{\mathbf{h}}_t], \tag{2.11}$$

Penggabungan ($\oplus$) kedua vektor ini memungkinkan model menangkap konteks karakter domain secara utuh, baik dari awalan (*prefix*) maupun akhiran (*suffix*), yang krusial untuk mendeteksi pola DGA yang kompleks [11].

2.7.3 BiLSTM dengan Fitur *Handcrafted*

Selain BiLSTM murni berbasis sekuens, penelitian ini juga menguji pendekatan hibrida dengan menggabungkan representasi BiLSTM dan fitur *handcrafted*. Representasi sekuens hasil BiLSTM dinyatakan sebagai $\mathbf{h}$, sedangkan vektor fitur *handcrafted* dinyatakan sebagai $\mathbf{f}$. Kedua vektor digabungkan melalui konkatenasi, lalu dipetakan menjadi probabilitas kelas biner menggunakan fungsi sigmoid:

$$\mathbf{z} = [\mathbf{h}\ \mathbf{f}], \qquad \hat{p}(y{=}1|\mathbf{z}) = \sigma(\mathbf{W}\mathbf{z} + b). \tag{2.12}$$

Pada penelitian ini, $y{=}1$ merepresentasikan kelas DGA dan $y{=}0$ merepresentasikan kelas legit.

2.8 *Feature Engineering* dalam Deteksi DGA

2.8.1 Konsep dan Relevansi

Feature engineering adalah proses transformasi data mentah menjadi representasi numerik yang lebih informatif guna meningkatkan kinerja model pembelajaran mesin [24]. Dalam deteksi DGA, kualitas fitur sangat menentukan kemampuan model dalam membedakan domain berbahaya (*malicious*) dan domain sah (*legit*/*benign*), terutama ketika menggunakan algoritma klasik seperti Random Forest [8].

2.8.2 Ekstraksi Fitur (*Feature Extraction*)

Penelitian ini menerapkan dua metode ekstraksi fitur utama:

1. Fitur statistik leksikal: fitur yang dihitung secara manual (*handcrafted*), seperti rasio vokal, rasio angka, dan tingkat keacakan string. Salah satu metrik penting adalah entropi Shannon (*Shannon entropy*), yang mengukur ketidakpastian informasi dalam string domain:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i), \tag{2.13}$$

dengan $p(x_i)$ adalah probabilitas kemunculan karakter x_i. Domain DGA umumnya memiliki entropi lebih tinggi dibanding domain bahasa alami [8].

2. TF–IDF *n-gram* karakter: penelitian ini menggunakan TF–IDF pada *n-gram* karakter untuk merepresentasikan pola substring pada SLD. Bobot TF–IDF mengikuti skema *smoothed idf* yang umum digunakan:

$$tf(t,d) = \text{frekuensi kemunculan } t \text{ pada domain } d, \tag{2.14}$$

$$idf(t) = \log\left(\frac{1+N}{1+df(t)}\right) + 1, \tag{2.15}$$

$$tfidf(t,d) = tf(t,d) \cdot idf(t), \tag{2.16}$$

dengan N adalah jumlah total sampel domain dan $df(t)$ adalah jumlah sampel yang memuat t. Selanjutnya, vektor TF–IDF tiap domain dinormalisasi menggunakan norma L_2 agar skala antar sampel lebih sebanding:

$$\mathbf{v}_d = \frac{\mathbf{tfidf}_d}{\|\mathbf{tfidf}_d\|_2}. \tag{2.17}$$

Contoh pembentukan (2,3)-gram dan hasil final TF–IDF: Misalkan SLD contoh adalah `q7xk9a`. Maka *2-gram* dan *3-gram* yang terbentuk adalah:

Tabel 2.2. Contoh *(2,3)-gram* dari sebuah SLD

SLD	***2-gram***	***3-gram***
`q7xk9a`	`q7, 7x, xk, k9, 9a`	`q7x, 7xk, xk9, k9a`

Jika kosakata global TF–IDF memuat *n-gram* tersebut, maka satu domain direpresentasikan sebagai vektor jarang (*sparse vector*). Sebagai ilustrasi, setelah pembobotan TF–IDF dan normalisasi L_2, domain `q7xk9a` dapat

memiliki pasangan fitur–bobot (nilai bobot ilustratif) seperti: {(xk:0.38), (k9:0.34), (q7:0.31), (7xk:0.29), (xk9:0.27), ...}. Vektor inilah yang menjadi masukan model berbasis fitur seperti *Random Forest*.

Teknik ini efektif menangkap pola substring unik pada SLD, sehingga dapat membantu membedakan domain DGA yang cenderung acak dari domain legit yang lebih mengikuti pola bahasa alami [8].

2.8.3 Standardisasi Fitur

Sebagian fitur *handcrafted* pada penelitian ini distandardisasi agar berada pada skala yang sebanding ketika digabung dengan fitur lain. Standardisasi dilakukan menggunakan *z-score*:

$$z = \frac{x - \mu}{\sigma}, \tag{2.18}$$

dengan μ adalah rata-rata fitur dan σ adalah simpangan baku fitur. Langkah ini membantu mencegah dominasi fitur tertentu hanya karena perbedaan skala nilai, terutama saat fitur *handcrafted* digabung dengan representasi lain.

2.8.4 Penambahan Fitur Eksternal (*Feature Addition*)

Selain fitur intrinsik, penelitian ini memperkaya data dengan fitur eksternal berbasis pengetahuan (*knowledge-based*). Hal ini meliputi penggunaan daftar reputasi domain populer (seperti Tranco) untuk memvalidasi kewajaran pola n-gram, serta kamus linguistik (English Wordlist) untuk mendeteksi keberadaan kata bermakna dalam nama domain [9]. Integrasi fitur eksternal ini memberikan konteks tambahan yang tidak tersedia hanya dari analisis string semata [9].

2.8.5 Seleksi Fitur (*Feature Selection*)

Tahap seleksi fitur bertujuan mereduksi dimensi data dengan membuang fitur yang redundan atau tidak relevan. Analisis korelasi antar fitur sering digunakan untuk mengidentifikasi multikolinearitas, sehingga model yang dihasilkan menjadi lebih efisien secara komputasi dan memiliki kemampuan generalisasi yang lebih baik [8].

2.9 Evaluasi Kinerja Model

Evaluasi model klasifikasi biner didasarkan pada matriks kebingungan (*Confusion Matrix*) yang mencatat jumlah prediksi benar (*True Positive, True Negative*) sedangkan prediksi salah (*False Positive, False Negative*) [24].

2.9.1 Metrik Utama

Pada penelitian ini, kelas positif didefinisikan sebagai domain DGA ($y = 1$) dan kelas negatif sebagai domain *legit* ($y = 0$). Dengan definisi tersebut: *true positive* (TP) adalah domain DGA yang diprediksi DGA, *true negative* (TN) adalah domain *legit* yang diprediksi *legit*, *false positive* (FP) adalah domain *legit* yang diprediksi DGA, dan *false negative* (FN) adalah domain DGA yang diprediksi *legit*. Istilah *recall* identik dengan sensitivitas (*true positive rate*/TPR). Untuk konsistensi, laporan ini menggunakan istilah *recall*.

- Akurasi (*accuracy*): proporsi total prediksi yang benar.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.19}$$

- Presisi (*precision*): tingkat kepercayaan prediksi DGA, penting untuk menekan *false positive*.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.20}$$

- *Recall* (sensitivitas/TPR): kemampuan model menangkap seluruh domain DGA, penting untuk menekan *false negative*.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.21}$$

- *F1-score*: rata-rata harmonik presisi dan *recall*, berguna saat data tidak seimbang.

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.22}$$

2.9.2 ROC-AUC

Receiver Operating Characteristic - Area Under Curve (ROC-AUC) mengukur kemampuan diskriminasi model pada berbagai ambang batas klasifikasi. Nilai AUC 1.0 menunjukkan prediksi sempurna, sedangkan 0.5 menunjukkan prediksi acak [24].

Kurva ROC memplot *true positive rate* (TPR) terhadap *false positive rate* (FPR) pada berbagai ambang keputusan:

$$TPR = \frac{TP}{TP+FN}, \qquad FPR = \frac{FP}{FP+TN}. \tag{2.23}$$

Pada penelitian ini, nilai AUC dihitung menggunakan skor probabilitas keluaran model.

2.10 Validasi Silang (*Cross-Validation*)

Untuk menjamin validitas hasil pengujian, penelitian ini menggunakan 5-fold *cross-validation*. Secara umum, dataset dibagi menjadi K bagian (*fold*), lalu model dilatih menggunakan $K-1$ *fold* dan diuji pada 1 *fold* secara bergantian hingga setiap *fold* pernah menjadi data uji. Pada variasi *Stratified K-Fold*, pembagian *fold* menjaga proporsi kelas pada setiap *fold* agar tetap konsisten. Pada penelitian ini, pendekatan *K-Fold* digunakan pada eksperimen Random Forest, sedangkan *Stratified K-Fold* digunakan pada eksperimen BiLSTM [25]. Skor akhir diperoleh dari rata-rata performa seluruh iterasi:

$$\overline{M} = \frac{1}{K}\sum_{k=1}^{K} M_k. \tag{2.24}$$

Metode ini memberikan estimasi kinerja yang lebih objektif dan bias yang lebih rendah dibandingkan metode *hold-out* tunggal, terutama dalam skenario keamanan siber di mana distribusi ancaman dapat bervariasi [26].

2.11 Konsep Dasar XAI (Interpretabilitas)

Model pembelajaran mesin modern yang akurat sering bersifat *black-box*, artinya alasan di balik suatu prediksi sulit dijelaskan secara langsung, terutama pada model kompleks seperti *deep learning* dan beberapa pendekatan *ensemble* [27, 28]. Kondisi ini menjadi isu penting pada keamanan siber, karena

keputusan seperti mengklasifikasikan sebuah nama domain sebagai DGA atau legit berpotensi memicu tindakan mitigasi (misalnya pemblokiran), sehingga perlu dapat dipertanggungjawabkan kepada analis dan pemangku kepentingan [29]. Oleh karena itu, *Explainable AI* (XAI) digunakan untuk meningkatkan transparansi dan kepercayaan terhadap model, tanpa mengubah tujuan utama model sebagai pengklasifikasi [28].

Penjelasan XAI dapat dibedakan menjadi berskala global dan lokal [30]. Penjelasan global bertujuan menggambarkan pola umum perilaku model pada sekumpulan data, misalnya fitur apa yang paling dominan memengaruhi keputusan model. Sebaliknya, penjelasan lokal menjawab pertanyaan mengapa satu sampel tertentu diprediksi sebagai DGA atau legit, sehingga membantu analis melakukan penelusuran keputusan pada kasus individual [30]. Dalam penelitian ini, kebutuhan tersebut selaras dengan tujuan interpretasi: memahami sinyal apa yang dipelajari model Random Forest dari vektor fitur, serta bagaimana model BiLSTM memanfaatkan pola urutan karakter pada SLD.

2.12 SHapley Additive exPlanations

SHAP (*SHapley Additive exPlanations*) merupakan metode XAI yang membangun penjelasan prediksi berbasis konsep nilai Shapley dari teori permainan kooperatif [30]. Pada kerangka ini, setiap fitur diperlakukan sebagai "pemain" yang berkontribusi terhadap keluaran model, sedangkan prediksi model dipandang sebagai "payout" yang perlu dibagi secara adil sesuai kontribusi masing-masing fitur [30]. Nilai kontribusi fitur ke-j (Shapley value) dihitung sebagai rata-rata kontribusi marginal fitur tersebut di semua kemungkinan urutan penambahan fitur ke dalam suatu himpunan fitur:

$$\phi_j = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!\,(M-|S|-1)!}{M!} \left[f_{S\cup\{j\}}(x_{S\cup\{j\}}) - f_S(x_S) \right], \tag{2.25}$$

dengan F adalah himpunan seluruh fitur, $M = |F|$, S adalah subset fitur yang tidak memuat fitur j, dan $f_S(x_S)$ adalah keluaran model ketika hanya fitur dalam S yang dianggap tersedia [30].

SHAP juga menggunakan bentuk penjelasan aditif, yaitu prediksi untuk sebuah sampel x direpresentasikan sebagai nilai dasar (baseline) ditambah

penjumlahan kontribusi tiap fitur:

$$f(x) = f_0 + \sum_{j=1}^{M} \phi_j, \tag{2.26}$$

sehingga total kontribusi seluruh fitur menjelaskan selisih antara prediksi sampel tersebut dan baseline model [30]. Sifat ini penting karena membuat penjelasan SHAP konsisten: kontribusi fitur dapat dibandingkan antar sampel, dan dapat diringkas menjadi penjelasan global (misalnya dengan rata-rata nilai absolut SHAP) maupun lokal (untuk satu sampel tertentu).

2.13 Varian SHAP pada Random Forest dan BiLSTM dalam Penelitian Ini

Implementasi SHAP bergantung pada jenis model yang dijelaskan [30]. Pada penelitian ini, model Random Forest bekerja pada masukan vektor fitur (misalnya fitur leksikal dan fitur berbasis *n-gram*). Untuk model berbasis pohon seperti Random Forest, SHAP menyediakan pendekatan yang memanfaatkan struktur pohon untuk menghitung kontribusi fitur secara efisien melalui *TreeExplainer* [30]. Dengan demikian, penjelasan global dapat diperoleh dengan merangkum nilai SHAP lintas data (misalnya peringkat fitur paling berpengaruh), dan penjelasan lokal dapat digunakan untuk menelusuri alasan sebuah domain diprediksi sebagai DGA atau legit.

Berbeda dengan itu, model BiLSTM memproses masukan sekuens karakter (SLD yang telah ditokenisasi dan dipadatkan), sehingga model tidak memiliki struktur pohon yang dapat dieksploitasi secara langsung. Untuk model seperti ini, pendekatan yang digunakan adalah *KernelExplainer* (Kernel SHAP), yaitu metode *model-agnostic* yang mengaproksimasi nilai Shapley melalui sampling berbagai kombinasi fitur yang diaktifkan/dinonaktifkan [30]. Pada praktiknya, Kernel SHAP memerlukan sekumpulan data latar (*background*) sebagai representasi baseline; pada implementasi penelitian ini, baseline tersebut dibentuk dari sampel latar yang ringkas dan representatif (misalnya melalui peringkasan berbasis klaster) agar perhitungan tetap efisien pada data berdimensi tinggi. Dalam konteks sekuens, fitur dapat dipandang sebagai posisi token/karakter (atau representasi numeriknya), lalu kontribusi tiap posisi diestimasi dari perubahan keluaran model ketika sebagian input diperturbasi [30]. Hasilnya dapat dirangkum menjadi penjelasan global maupun lokal sehingga relevan untuk tujuan penelitian ini, yaitu memahami pola apa yang dipakai BiLSTM murni dan BiLSTM hibrida ketika membedakan domain

DGA dan domain legit.

2.14 Sumber Data dan Basis Pengetahuan Eksternal

Dalam penelitian ini, dataset disusun dari unifikasi berbagai sumber sekunder untuk menjamin keragaman pola serangan dan validitas lalu lintas normal. Sumber data dikategorikan menjadi dua kelompok fungsi:

1. Sumber sampel domain (*sample source*) yaitu:
    - Kelas DGA (*malicious*): sampel domain DGA dihimpun dari UMUDGA [31], Data Driven Security [32], ExtraHop [33], serta Kaggle [34].
    - Kelas normal (*legit*): penelitian ini menggunakan arsip Alexa Top-1M. Penggunaan daftar domain populer/*trusted* merupakan praktik umum untuk menekan *false positive* [34].
2. Basis pengetahuan eksternal (*feature source*) sumber eksternal digunakan sebagai referensi (*lookup table*) dalam proses rekayasa fitur statis yaitu:
    - Tranco top list: daftar *Tranco* digunakan sebagai pembanding reputasi yang lebih modern dan resisten terhadap manipulasi peringkat [35]. Dalam penelitian ini, Tranco berfungsi sebagai referensi untuk fitur statistik (misalnya menghitung frekuensi *n-gram* domain populer).
    - English wordlist: kamus kosakata bahasa Inggris (bersumber dari dataset pendukung UMUDGA [31]) digunakan untuk mendeteksi keberadaan kata bermakna (*dictionary words*), terutama untuk membedakan DGA berbasis kamus dengan domain acak.